Ryerson University

AMiR Lab
Connecting the dots

IBM

LCaaS:
Immutable Log Storage as a Service

William Pourmajidi
Department of Computer Science
Ryerson University
Toronto, Canada

Lei Zhang
Department of Computer Science
Ryerson University
Toronto, Canada

John Steinbacher
IBM Canada Lab
Toronto, Canada

Tony Erwin
IBM Watson and Cloud Platform,
Austin, USA

Andriy Miranskyy
Department of Computer Science
Ryerson University
Toronto, Canada

Agenda

- Abstract
- Introduction
- Design of LCaaS
- Evaluation
- Conclusion and Future Work
- Q&A

Abstract

Importance of Logs

- **During normal operation:** Logs are important to technical dept
- **At the time of crisis:** Logs are important to everyone
- **SLA Compliance:** Performance, Availability reflected in logs
- **Party at fault:** May be motivated to adjust, remove, or change current logs

Proposed Solution

- **Logchain as a Service(LCaaS):** Using immutability of blockchain to prevent log tampering while addressing blockchain scalability issues

Introduction

Cloud Computing Logs

- Evidential documents [1]: Authentic, Admissible
- Used for computer forensic investigations [2]
- **Tamper Motivation:** the desire of one or more of the parties involved in a Cloud platform to access logs and to tamper them
- **Tampering forms:** Adding, removing, or manipulating files/logs

[1] R. Accorsi, “Log data as digital evidence: What secure logging protocols , have to offer?” in 33rd Annual IEEE Int. Computer Software and Applications Conference, COMPSAC’09., vol. 2. IEEE, 2009, pp. 398–403.
[2] D. Reilly, C. Wren, and T. Berry, “Cloud computing: Forensic challenges for law enforcement,” in Int. Conference for Internet Technology and Secured Transactions (ICITST). IEEE, 2010, pp. 1–7.

Introduction

Private Cloud Tamper Motivations

- Internal team tamper motivation
- **Example:** Adjusting the logs that shows the failure of IT team to backup Tier-0 data to pass the blame to the backup platform

Community Cloud Tamper Motivations

- One or more parties deviate from their assigned tasks
- **Example:** Adjusting the logs that shows the failure is caused by a party (let's say party A) and replace it with logs that shows another party (let's say party B) is the party at fault

Introduction

Public Cloud Tamper Motivations

- Clients trust Cloud providers:
  - Without access to bare-metal servers and their logs
  - Without access to physical networks and their logs
- **Example:** If auto-scaling configuration fails, how does the client get access to the actual auto-scaling logs?

Monitoring Cloud Environment

- Monitoring is usually managed by the Cloud providers

Introduction

Our Proposed Solution

Logchain as a Service (LCaaS): a temper-proof log storage system that:

- Uses the immutability of blockchain
- Saves data in a hierarchical , scalable ledger
- Acts as a service (Logchain as a Service - LCaaS)
- Provides an API for submission and verification

Main Challenges

- Integration points
- Scalability of blockchains

Design of LCaaS

Proposed Solution: Logchain

- A hierarchical ledger (multi-level)
- Addresses current blockchain limitation by:
  - Segmenting a portion of blockchain (Circled blockchains)
  - Locking it in a higher-level blockchain
- **Faster Proof:** Validating the integrity of a block of higher-level blockchain, is the proof that all blocks in the lower-level blockchain were not tampered

Design of LCaaS

Common Key Components of Blockchains

Genesis Blocks (GB):

- The first block of any blockchain and has null *data* element
- Its index and *previous_hash* are set to zero

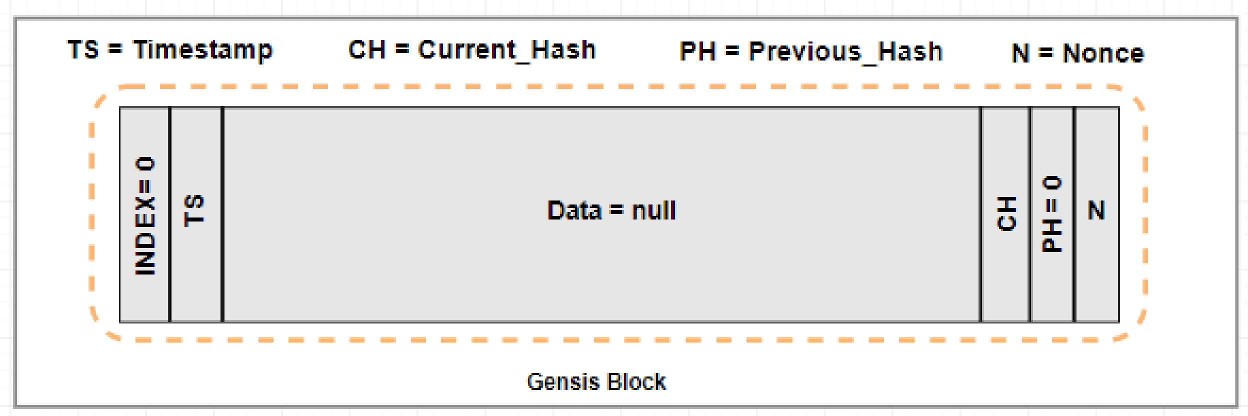


Figure 1: Genesis Block and its elements

Design of LCaaS

Common Key Components of Blockchains

Blocks (Data Blocks):

- Atomic unit of storage
- Hash Binding: The *current_hash* of the nth block becomes the *previous_hash* of block *n + 1*

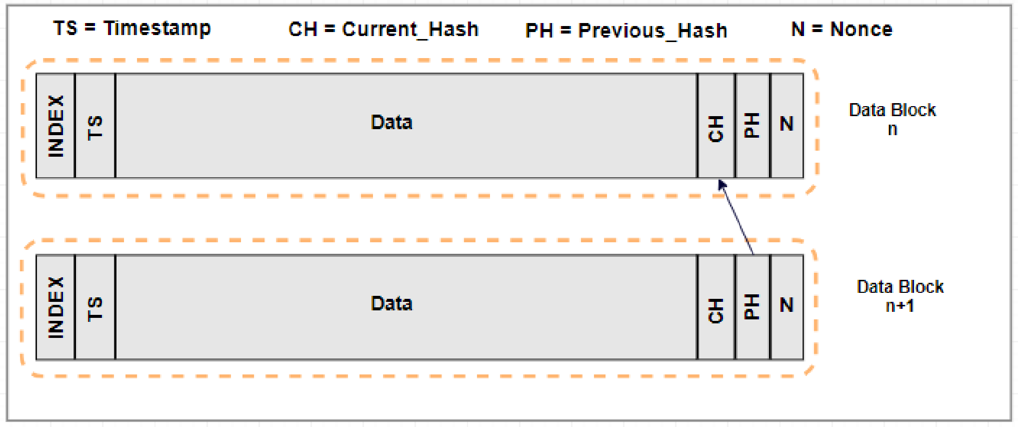


Figure 2: Data Block and Hash Binding

Design of LCaaS

Common Key Components of Blockchains

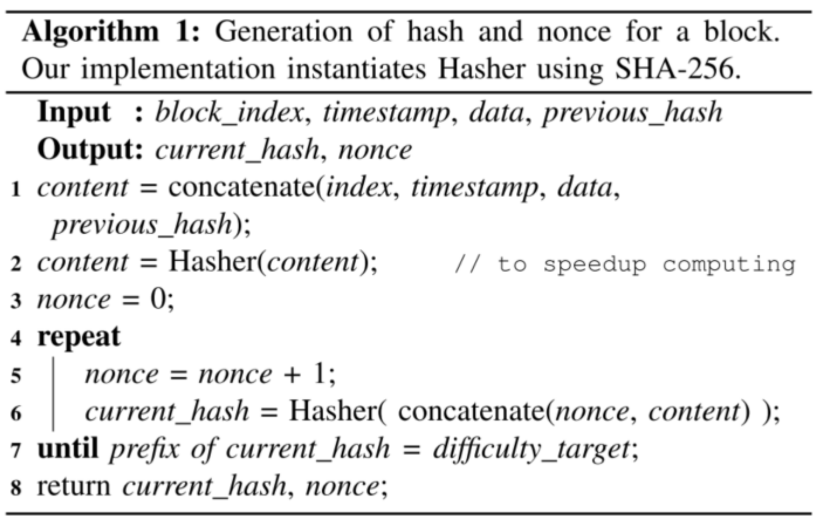
Blockchains:

- Blocks that are linked in a hash-binding relation construct a blockchain
- Tampering earlier blocks will break the link among all subsequent blocks

Design of LCaaS

Proof of Work (POW) and Difficulty Target

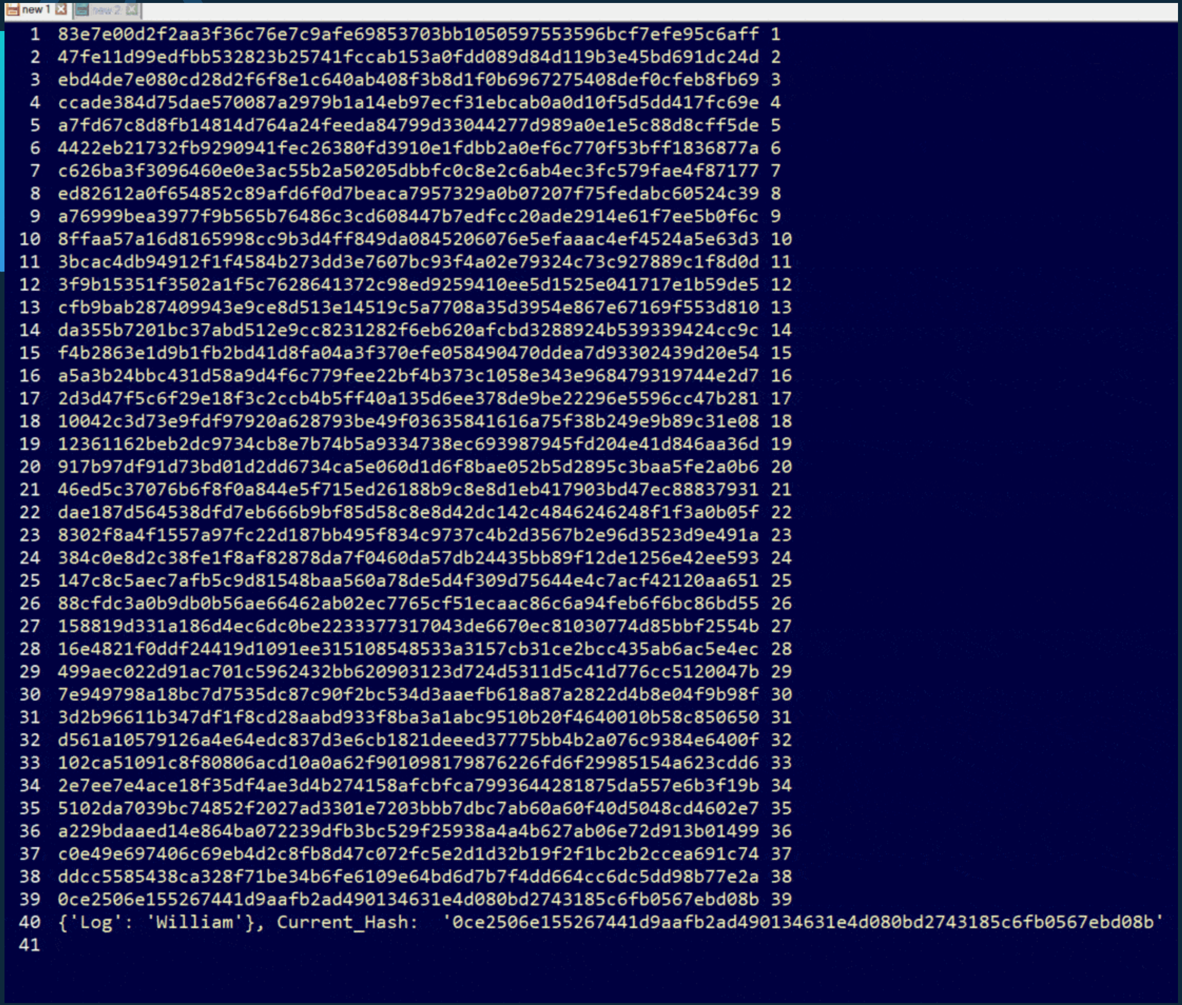
- **Difficulty target:** Number of zeros that must appear at the beginning of desired *current_hash.* It has major impact on computation time



Algorithm 1: Generation of hash and nonce for a block. Our implementation instantiates Hasher using SHA-256.

```
Input  : block_index, timestamp, data, previous_hash
Output: current_hash, nonce
content = concatenate(index, timestamp, data,
  previous_hash);
content = Hasher(content);        // to speedup computing
nonce = 0;
repeat
    nonce = nonce + 1;
    current_hash = Hasher( concatenate(nonce, content) );
until prefix of current_hash = difficulty_target;
return current_hash, nonce;
```

Algorithm1: *current_hash* and *nonce* generation



```
83e7e00d2f2aa3f36c76e7c9afe69853703bb1050597553596bcf7efe95c6aff 1
47fe11d99edfbb532823b25741fccab153a0fdd089d84d119b3e45bd691dc24d 2
ebd4de7e080cd28d2f6f8e1c640ab408f3b8d1f0b6967275408def0cfeb8fb69 3
ccade384d75dae570087a2979b1a14eb97ecf31ebcab0a0d10f5d5dd417fc69e 4
a7fd67c8d8fb14814d764a24feeda84799d33044277d989a0e1e5c88d8cff5de 5
4422eb21732fb9290941fec26380fd3910e1fdbb2a0ef6c770f53bff1836877a 6
c626ba3f3096460e0e3ac55b2a50205dbbfc0c8e2c6ab4ec3fc579fae4f87177 7
ed82612a0f654852c89afd6f0d7beaca7957329a0b07207f75fedabc60524c39 8
a76999bea3977f9b565b76486c3cd608447b7edfcc20ade2914e61f7ee5b0f6c 9
8ffaa57a16d8165998cc9b3d4ff849da0845206076e5efaaac4ef4524a5e63d3 10
3bcac4db94912f1f4584b273dd3e7607bc93f4a02e79324c73c927889c1f8d0d 11
3f9b15351f3502a1f5c7628641372c98ed9259410ee5d1525e041717e1b59de5 12
cfb9bab287409943e9ce8d513e14519c5a7708a35d3954e867e67169f553d810 13
da355b7201bc37abd512e9cc8231282f6eb620afcbd3288924b539339424cc9c 14
f4b2863e1d9b1fb2bd41d8fa04a3f370efe058490470ddea7d93302439d20e54 15
a5a3b24bbc431d58a9d4f6c779fee22bf4b373c1058e343e968479319744e2d7 16
2d3d47f5c6f29e18f3c2ccb4b5ff40a135d6ee378de9be22296e5596cc47b281 17
10042c3d73e9fdf97920a628793be49f03635841616a75f38b249e9b89c31e08 18
12361162beb2dc9734cb8e7b74b5a9334738ec693987945fd204e41d846aa36d 19
917b97df91d73bd01d2dd6734ca5e060d1d6f8bae052b5d2895c3baa5fe2a0b6 20
46ed5c37076b6f8f0a844e5f715ed26188b9c8e8d1eb417903bd47ec88837931 21
dae187d564538dfd7eb666b9bf85d58c8e8d42dc142c4846246248f1f3a0b05f 22
8302f8a4f1557a97fc22d187bb495f834c9737c4b2d3567b2e96d3523d9e491a 23
384c0e8d2c38fe1f8af82878da7f0460da57db24435bb89f12de1256e42ee593 24
147c8c5aec7afb5c9d81548baa560a78de5d4f309d75644e4c7acf42120aa651 25
88cfdc3a0b9db0b56ae66462ab02ec7765cf51ecaac86c6a94feb6f6bc86bd55 26
158819d331a186d4ec6dc0be2233377317043de6670ec81030774d85bbf2554b 27
16e4821f0ddf24419d1091ee315108548533a3157cb31ce2bcc435ab6ac5e4ec 28
499aec022d91ac701c5962432bb620903123d724d5311d5c41d776cc5120047b 29
7e949798a18bc7d7535dc87c90f2bc534d3aaefb618a87a2822d4b8e04f9b98f 30
3d2b96611b347df1f8cd28aabd933f8ba3a1abc9510b20f4640010b58c850650 31
d561a10579126a4e64edc837d3e6cb1821deeed37775bb4b2a076c9384e6400f 32
102ca51091c8f80806acd10a0a62f901098179876226fd6f29985154a623cdd6 33
2e7ee7e4ace18f35df4ae3d4b274158afcbfca7993644281875da557e6b3f19b 34
5102da7039bc74852f2027ad3301e7203bbb7dbc7ab60a60f40d5048cd4602e7 35
a229bdaaed14e864ba072239dfb3bc529f25938a4a4b627ab06e72d913b01499 36
c0e49e697406c69eb4d2c8fb8d47c072fc5e2d1d32b19f2f1bc2b2ccea691c74 37
ddcc5585438ca328f71be34b6fe6109e64bd6d7b7f4dd664cc6dc5dd98b77e2a 38
0ce2506e155267441d9aafb2ad490134631e4d080bd2743185c6fb0567ebd08b 39
{'Log': 'William'}, Current_Hash:  '0ce2506e155267441d9aafb2ad490134631e4d080bd2743185c6fb0567ebd08b'
```

Figure 3: Difficulty Target of one zero versus two zeros for {'Log': 'William'}

Design of LCaaS

Key Components of LCaaS

We extend Genesis Blocks by adding:

Absolute Genesis Blocks (AGB):

- added to first block of the first blockchain.
- Same characteristics of GB

Relative Genesis Block (RGB):

- added to the first block of any subsequent circled blockchain
- Its *current_hash* is set to *current_hash of the previous Terminal Block* (TBD)

Design of LCaaS

Key Components of LCaaS

Terminal Block (TB) :

- Similar to GB but added at the end of a blockchain
- Converts an “open” blockchain to a “closed” blockchain. We call it Circled blockchain (CB)
- Like a block but Its *data* element has additional details:
  - ***aggr_hash***: contains the hash of concatenated *current_hash* values of all blocks in CB (from AGB or RGB to block prior to TB)
  - *timestamp_ from, timestamp_ to , block_index_from , block_index_to* (useful for search API)

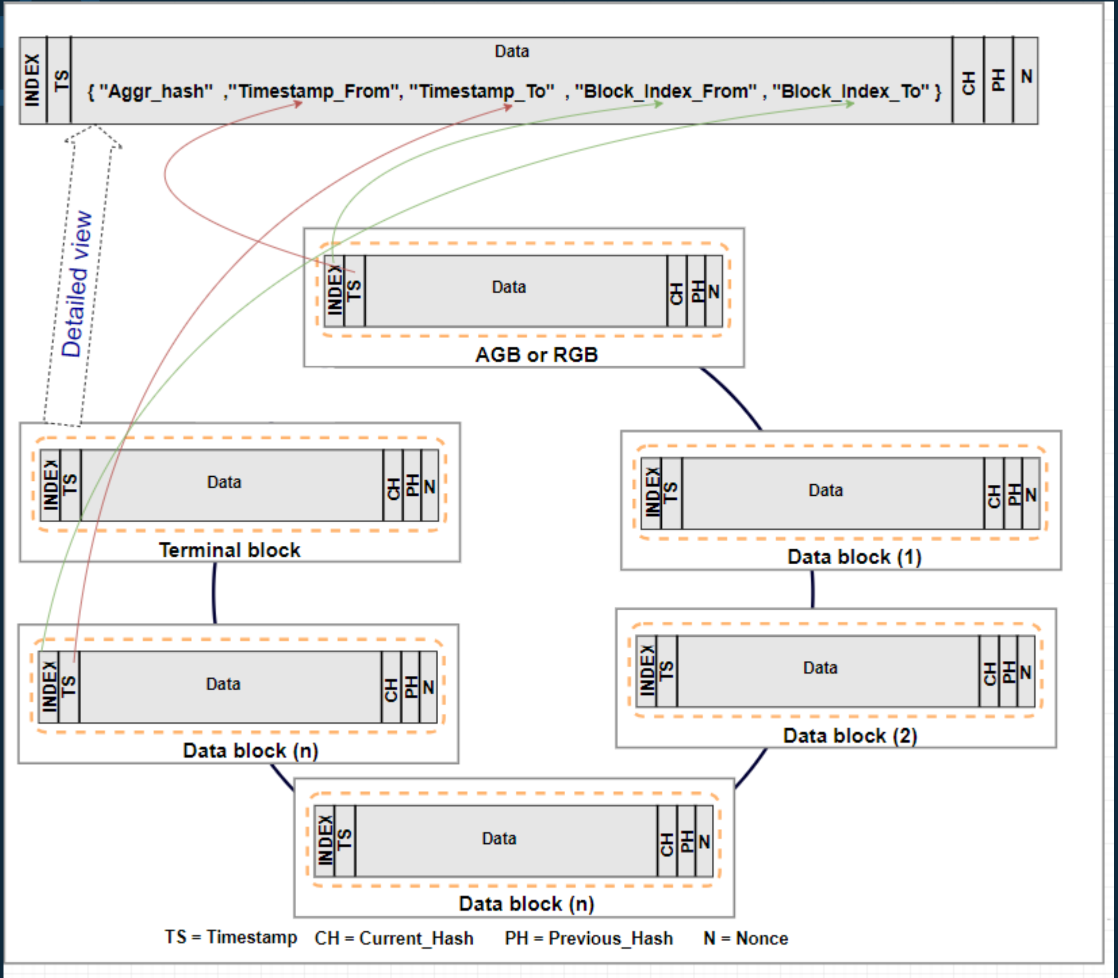

INDEX
TS
Data
{ "Aggr_hash" ,"Timestamp_From", "Timestamp_To" , "Block_Index_From" , "Block_Index_To" }
CH
PH
N
Detailed view
INDEX TS Data CH PH N
AGB or RGB
Terminal block
Data block (1)
Data block (2)
Data block (n)
Data block (n)
TS = Timestamp CH = Current_Hash PH = Previous_Hash N = Nonce

Figure 4: Terminal Block and Circled Blockchain

Design of LCaaS

Key Components of LCaaS

Circled Blockchain (CB): Starts with a Genesis Block and ends with a Terminal Block and is a closed-loop blockchain and can no longer accept any blocks

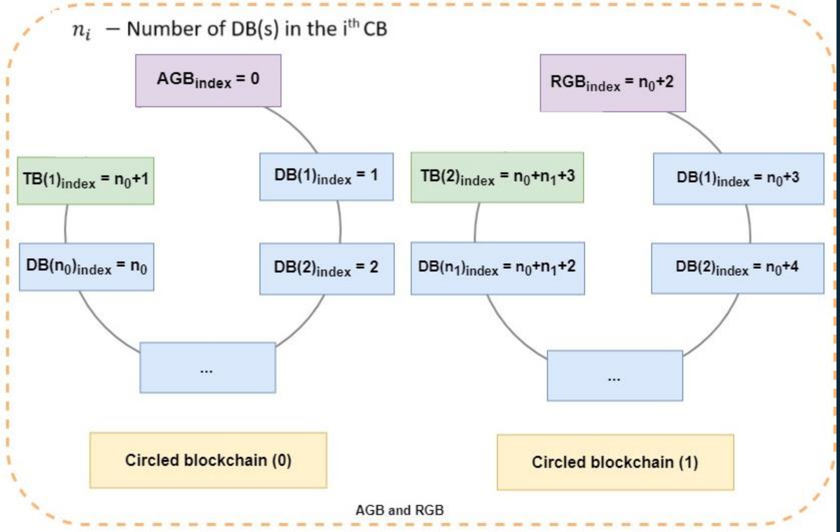


Figure 5: AGB and RGB in Circled Blockchain(0) and Circled Blockchain

Design of LCaaS

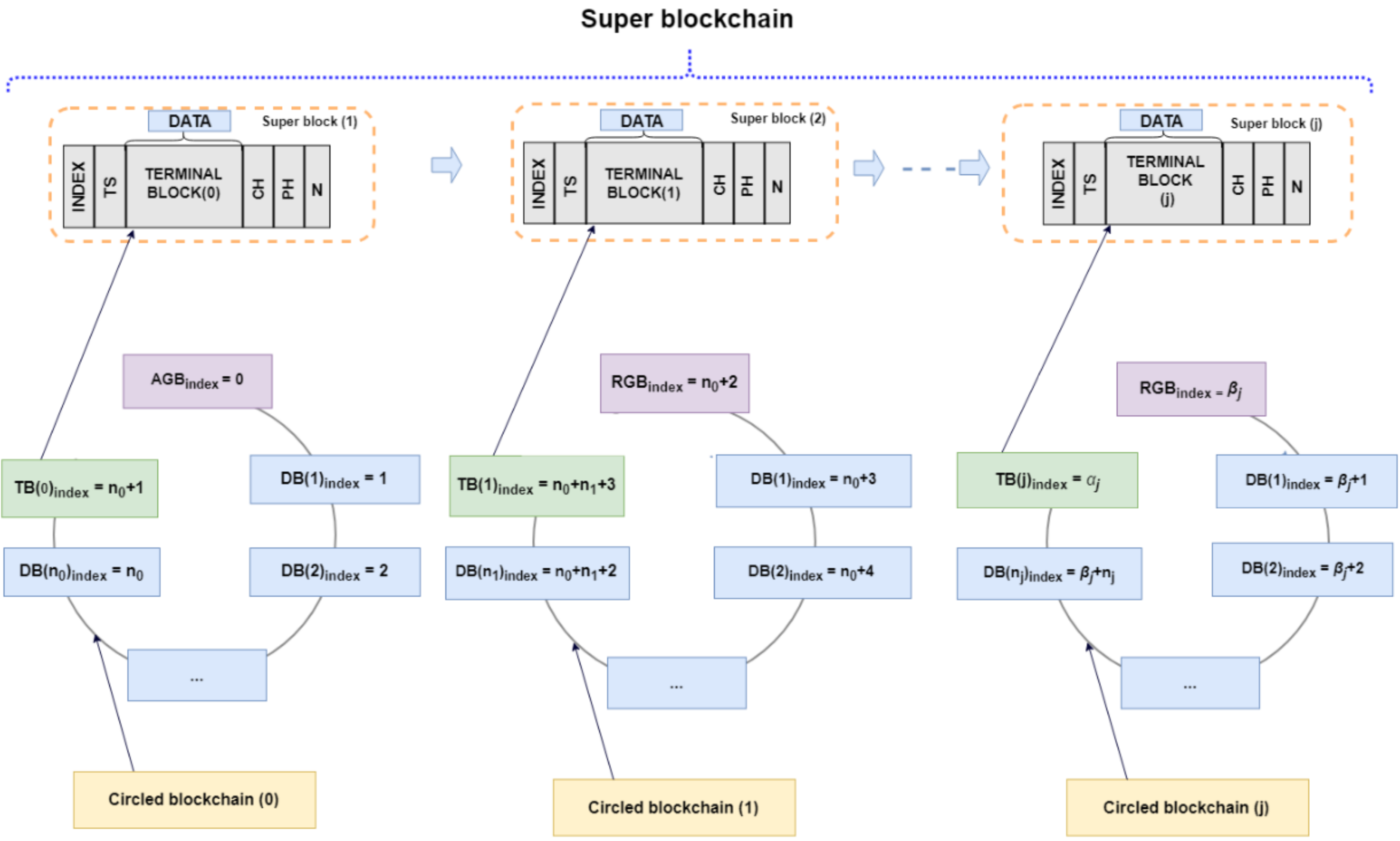
Key Components of LCaaS

Super Block (SB):

- Exhibits the same characteristic of a block
- Its data element stores all fields of a TB of a CB

Super Blockchain (SBC):

- A blockchain that its blocks are Super blocks
- Super blocks in a Super blockchain are chained by hash-binding relation



Super blockchain

DATA

Super block (1)

INDEX

TS

TERMINAL BLOCK(0)

CH

PH

N

DATA

Super block (2)

INDEX

TS

TERMINAL BLOCK(1)

CH

PH

N

DATA

Super block (j)

INDEX

TS

TERMINAL BLOCK (j)

CH

PH

N

$AGB_{index} = 0$

$TB(0)_{index} = n_0+1$

$DB(1)_{index} = 1$

$DB(n_0)_{index} = n_0$

$DB(2)_{index} = 2$

...

Circled blockchain (0)

$RGB_{index} = n_0+2$

$TB(1)_{index} = n_0+n_1+3$

$DB(1)_{index} = n_0+3$

$DB(n_1)_{index} = n_0+n_1+2$

$DB(2)_{index} = n_0+4$

...

Circled blockchain (1)

$RGB_{index} = \beta_j$

$TB(j)_{index} = \alpha_j$

$DB(1)_{index} = \beta_j+1$

$DB(n_j)_{index} = \beta_j+n_j$

$DB(2)_{index} = \beta_j+2$

...

Circled blockchain (j)

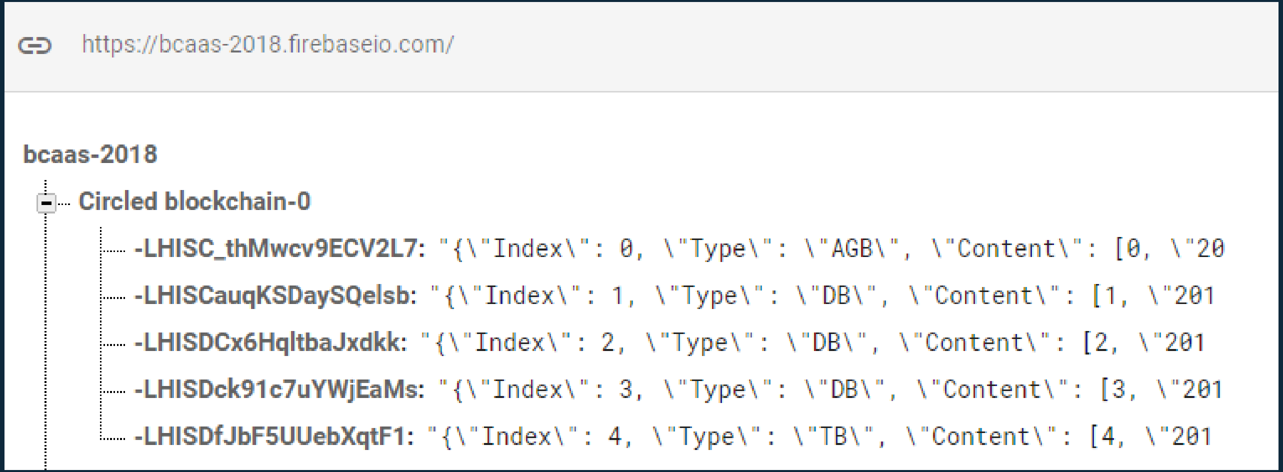
Figure 6: Two-level Hierarchy as implemented by Logchain

Design of LCaaS

Persistent Storage of Blocks

Google Firebase Real-time Database

- LCaaS stores generated blocks (of all types) on Google Firebase



https://bcaas-2018.firebaseio.com/

```
bcaas-2018
⊟ Circled blockchain-0
    -LHISC_thMwcv9ECV2L7: "{\"Index\": 0, \"Type\": \"AGB\", \"Content\": [0, \"20
    -LHISCauqKSDaySQelsb: "{\"Index\": 1, \"Type\": \"DB\", \"Content\": [1, \"201
    -LHISDCx6HqltbaJxdkk: "{\"Index\": 2, \"Type\": \"DB\", \"Content\": [2, \"201
    -LHISDck91c7uYWjEaMs: "{\"Index\": 3, \"Type\": \"DB\", \"Content\": [3, \"201
    -LHISDfJbF5UUebXqtF1: "{\"Index\": 4, \"Type\": \"TB\", \"Content\": [4, \"201
```

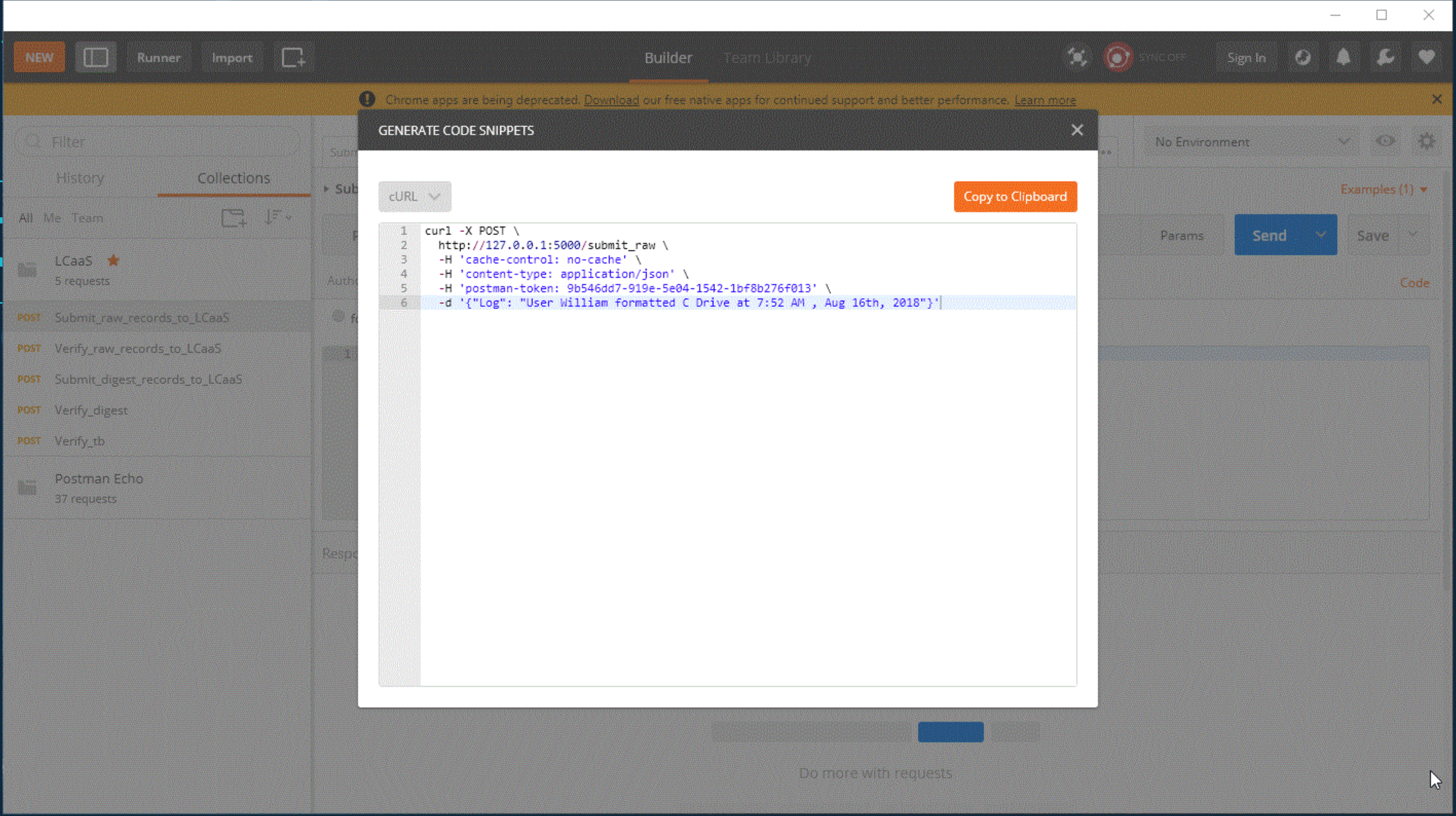
Figure 7: Storage of AGB, DB, and TB on Google Firebase

Design of LCaaS

LCaaS APIs

- Using REST and HTTP POST operation [22]
- **Submission Methods:**
  - *submit_raw()*
  - *sumbit_digest()*
- Returns
  - On success: {*status, data_block_details*}
  - On failure: {*error , details*}

[22] R. T. Fielding and R. N. Taylor, “Principled design of the modern web architecture,” ACM Trans. Internet Tech., vol. 2, no. 2, pp. 115–150



NEW
Runner
Import
Builder
Team Library
SYNC OFF
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
LCaaS
5 requests
POST Submit_raw_records_to_LCaaS
POST Verify_raw_records_to_LCaaS
POST Submit_digest_records_to_LCaaS
POST Verify_digest
POST Verify_tb
Postman Echo
37 requests
No Environment
Examples (1)
Params
Send
Save
Code
Do more with requests

GENERATE CODE SNIPPETS

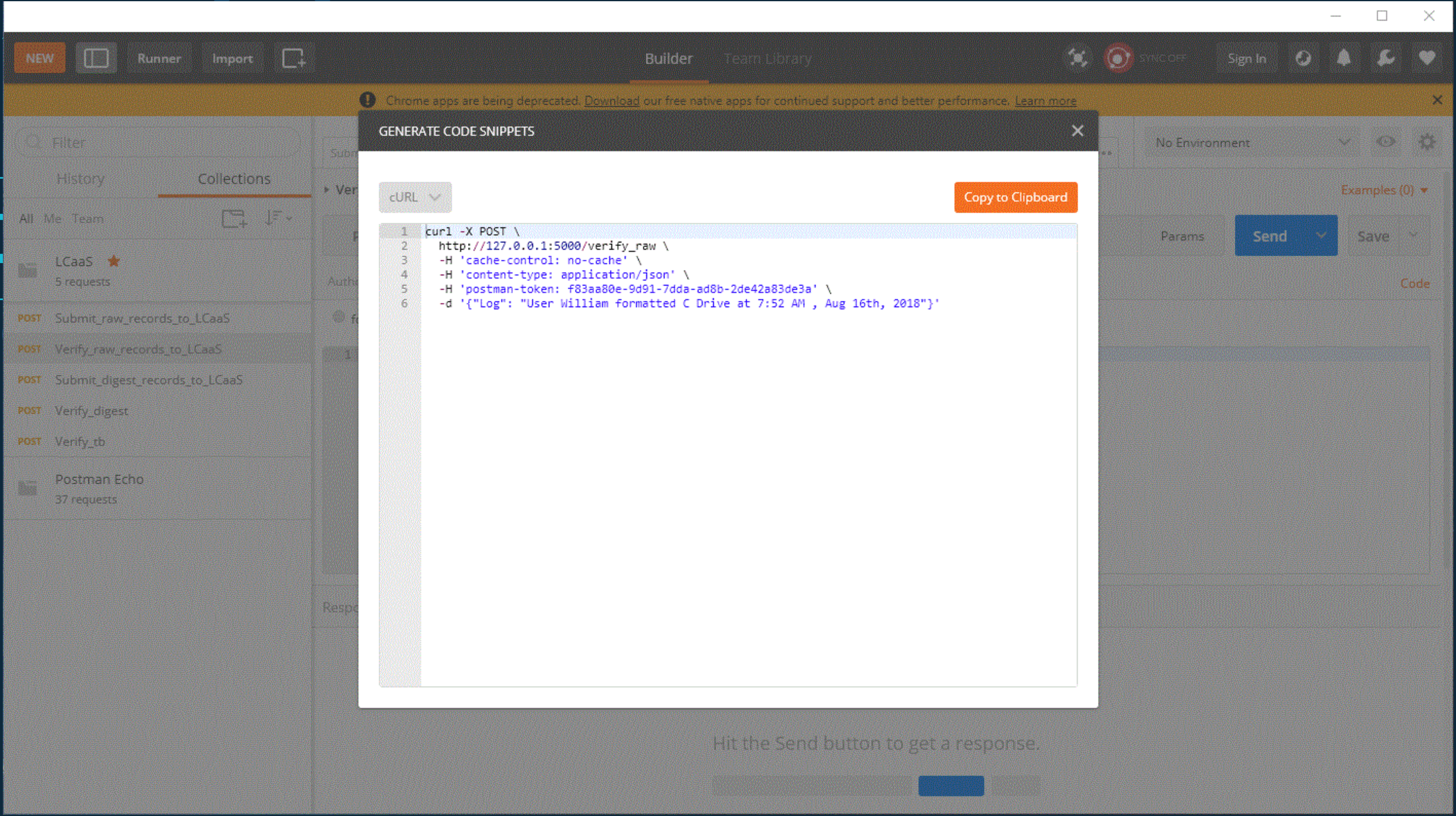
```
curl -X POST \
  http://127.0.0.1:5000/submit_raw \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: 9b546dd7-919e-5e04-1542-1bf8b276f013' \
  -d '{"Log": "User William formatted C Drive at 7:52 AM , Aug 16th, 2018"}'
```

Figure 8: An example of submit_raw Method

Design of LCaaS

LCaaS APIs

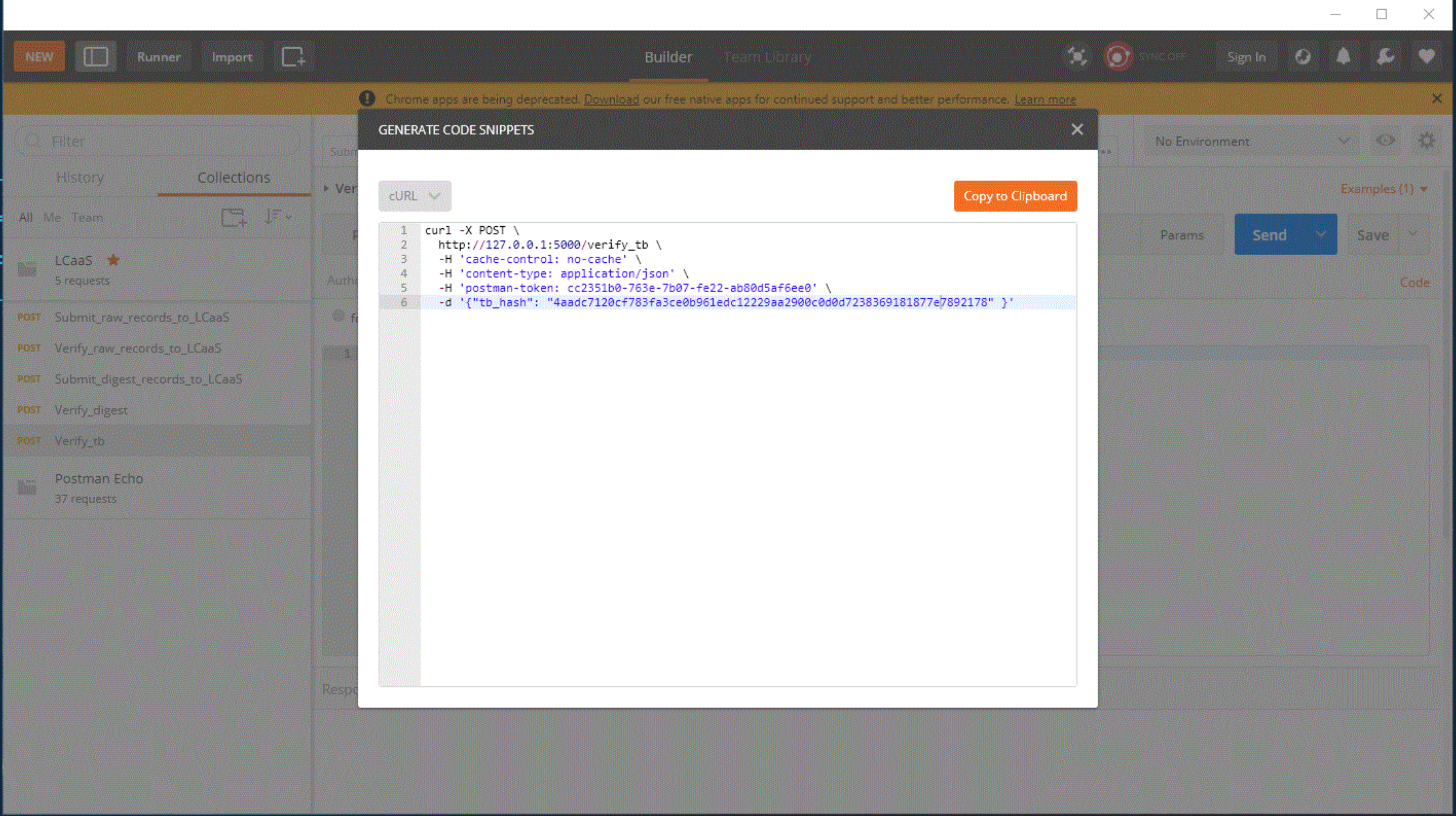
- **Verification Methods:**
  - *verify_raw()*
  - *verify_digest()*
  - *verify_tb()*
- Returns
  - On success: {*number of blocks that match(es)*
    and their timestamp(s)}
  - On failure: {*error , details*}



NEW
Runner
Import
Builder
Team Library
SYNC OFF
Sign In
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more
Filter
History
Collections
All Me Team
LCaaS
5 requests
POST Submit_raw_records_to_LCaaS
POST Verify_raw_records_to_LCaaS
POST Submit_digest_records_to_LCaaS
POST Verify_digest
POST Verify_tb
Postman Echo
37 requests
No Environment
Examples (0)
Params
Send
Save
Code
Hit the Send button to get a response.
GENERATE CODE SNIPPETS
cURL
Copy to Clipboard

```
curl -X POST \
  http://127.0.0.1:5000/verify_raw \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: f83aa80e-9d91-7dda-ad8b-2de42a83de3a' \
  -d '{"Log": "User William formatted C Drive at 7:52 AM , Aug 16th, 2018"}'
```

Figure 9: An example of verify_raw Method



```
curl -X POST \
  http://127.0.0.1:5000/verify_tb \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -H 'postman-token: cc2351b0-763e-7b07-fe22-ab80d5af6ee0' \
  -d '{"tb_hash": "4aadc7120cf783fa3ce0b961edc12229aa2900c0d0d7238369181877e7892178" }'
```

Figure 10: An example of verify_tb Method

Design of LCaaS

Applicability to other Blockchains

- No change in the structure of blockchain
- No change in the elements of blocks
- No changes in mining and hash binding

Hence, we believe that hierarchical ledger structure can be implemented on top of any blockchain

Design of LCaaS

LCaaS and Ethereum Integration

Ethereum as blockchain for LCaaS

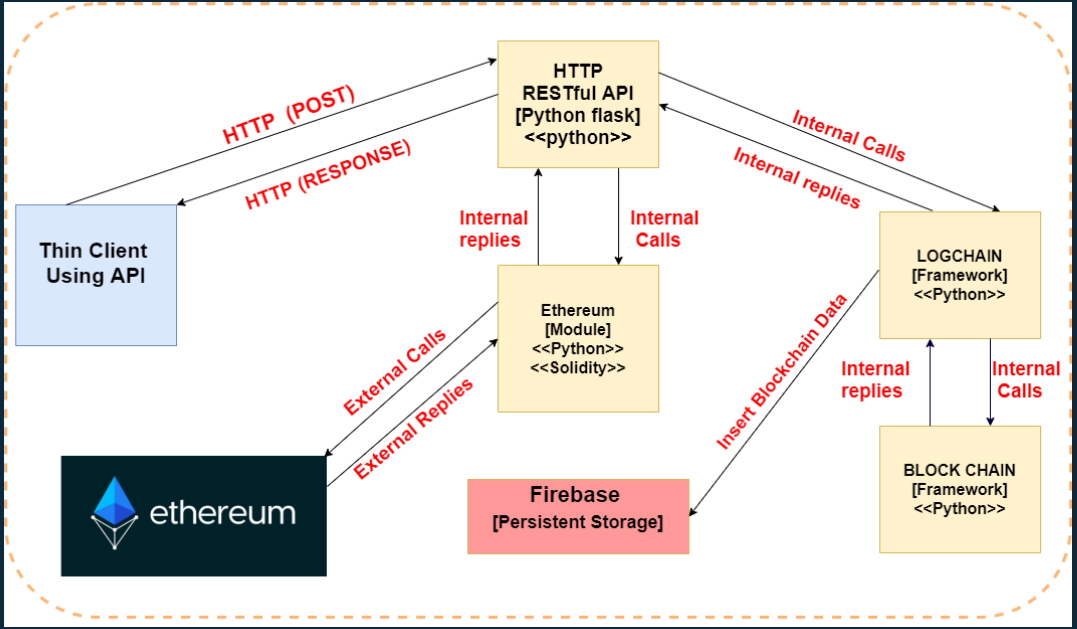


Figure 11: LCaaS and Ethereum Integration Point

Design of LCaaS

LCaaS and IBM Blockchain

Ethereum as blockchain for LCaaS

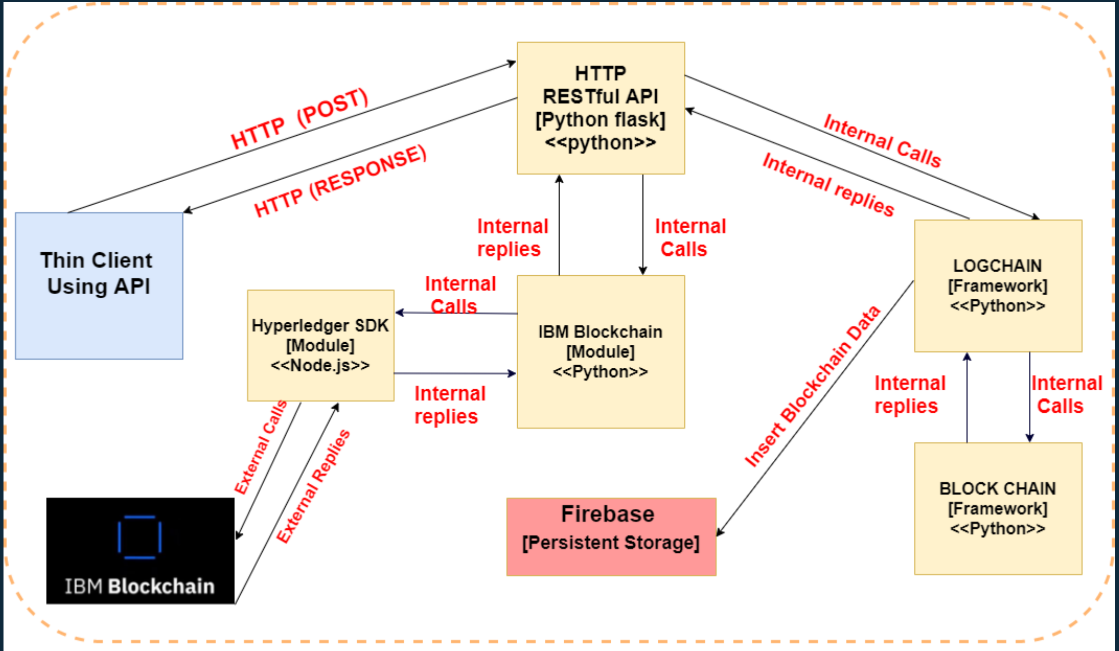
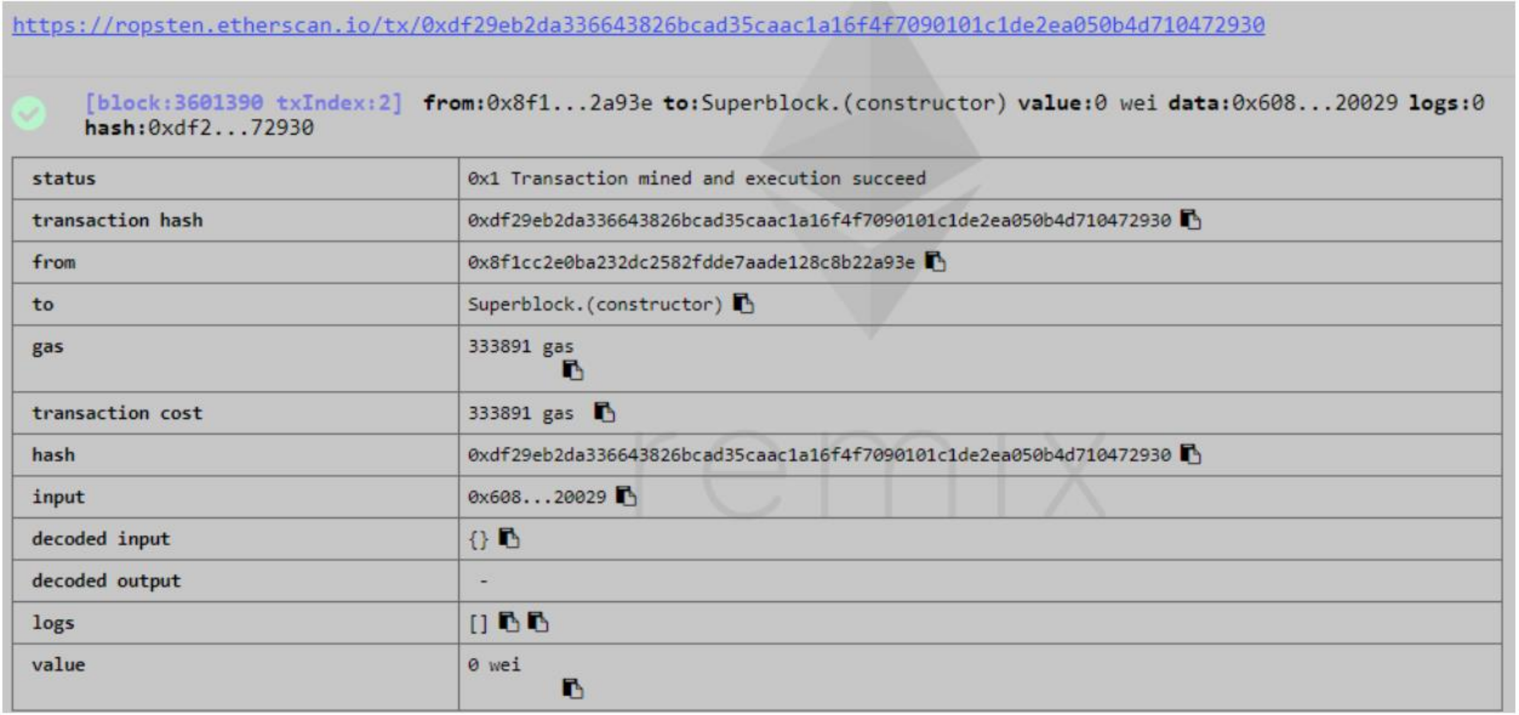


Figure 11: LCaaS and IBM Blockchain Integration Point

Evaluation

Publishing a Smart Contract



https://ropsten.etherscan.io/tx/0xdf29eb2da336643826bcad35caac1a16f4f7090101c1de2ea050b4d710472930

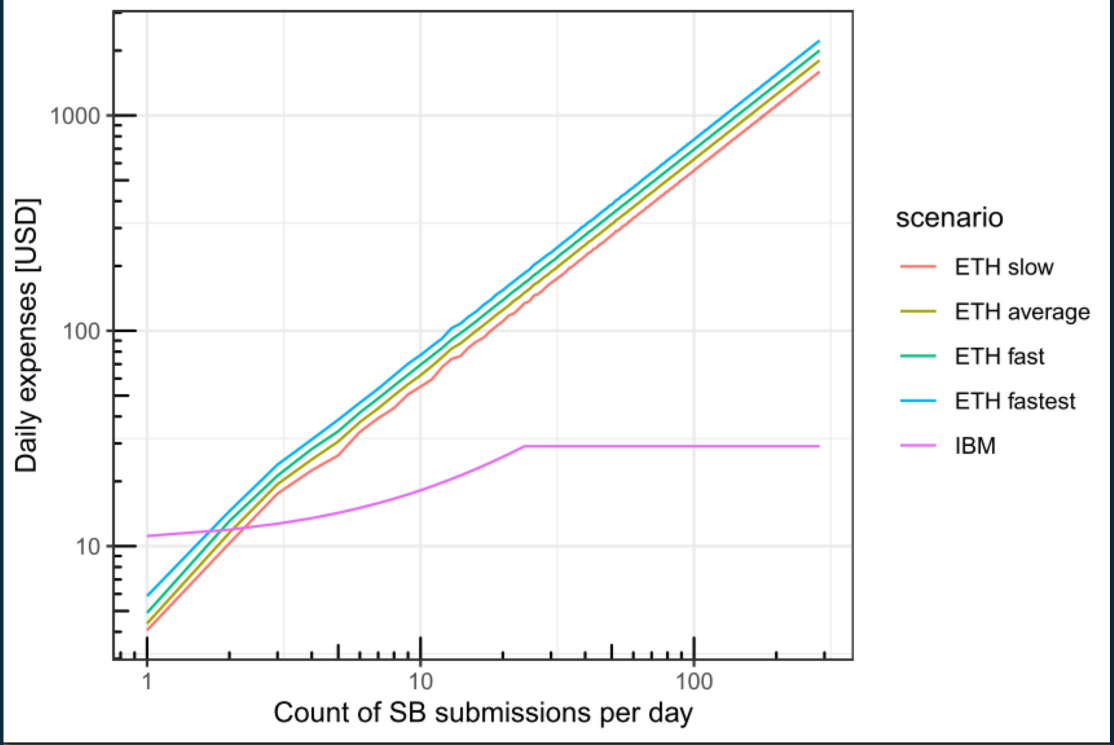
[block:3601390 txIndex:2] **from:**0x8f1...2a93e **to:**Superblock.(constructor) **value:**0 wei **data:**0x608...20029 **logs:**0 **hash:**0xdf2...72930

status	0x1 Transaction mined and execution succeed
transaction hash	0xdf29eb2da336643826bcad35caac1a16f4f7090101c1de2ea050b4d710472930
from	0x8f1cc2e0ba232dc2582fdde7aade128c8b22a93e
to	Superblock.(constructor)
gas	333891 gas
transaction cost	333891 gas
hash	0xdf29eb2da336643826bcad35caac1a16f4f7090101c1de2ea050b4d710472930
input	0x608...20029
decoded input	{}
decoded output	-
logs	[]
value	0 wei

Figure 12: Published Smart Contract on Ethereum Test Network (Ropsten)

Evaluation

Cost of Ownership



Conclusion

Primary objective: Trust issues among cloud participants
Solution: Use of Blockchain as log storage platform

Secondary objective: Reduce computational complexity of blockchain verification methods
Solution: Introduction of Logchain framework for hierarchical blockchains

Tertiary objective: Enhance accessibility of Logs
Solution: Introduction of LCaaS API

Our work published at

- 2018 IEEE 11th International Conference on Cloud Computing (CLOUD)
- 2019 IEEE/ACM 41st International Conference on Software (ICSE)
- 2020 Knowledge Management in the Development of Data-Intensive Systems (KMDD)
- 2021 IEEE Transactions on Services Computing

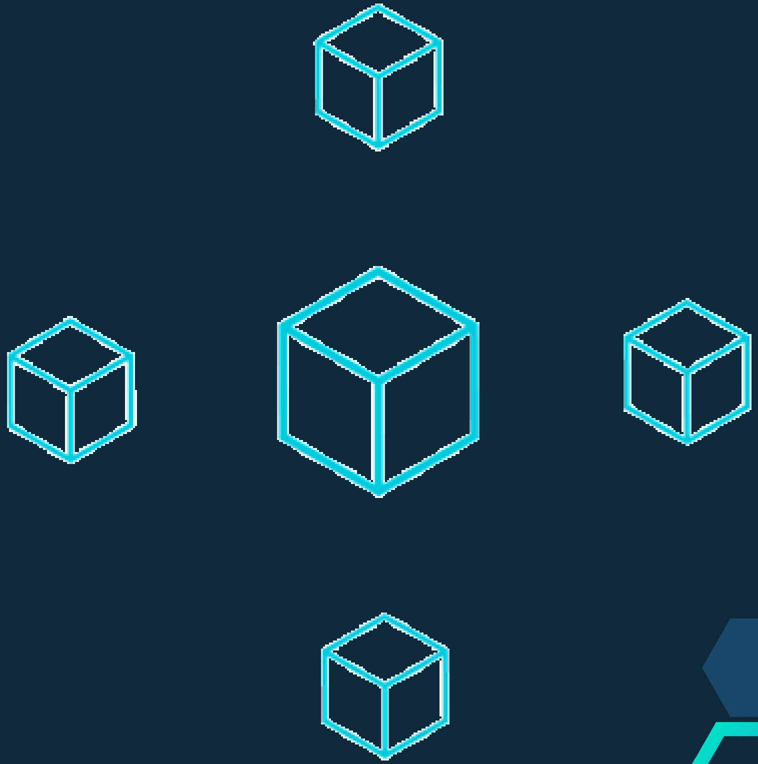
Git Hub https://github.com/WilliamPourmajidi/LCaaS

Future Work

LCaaS Improvements

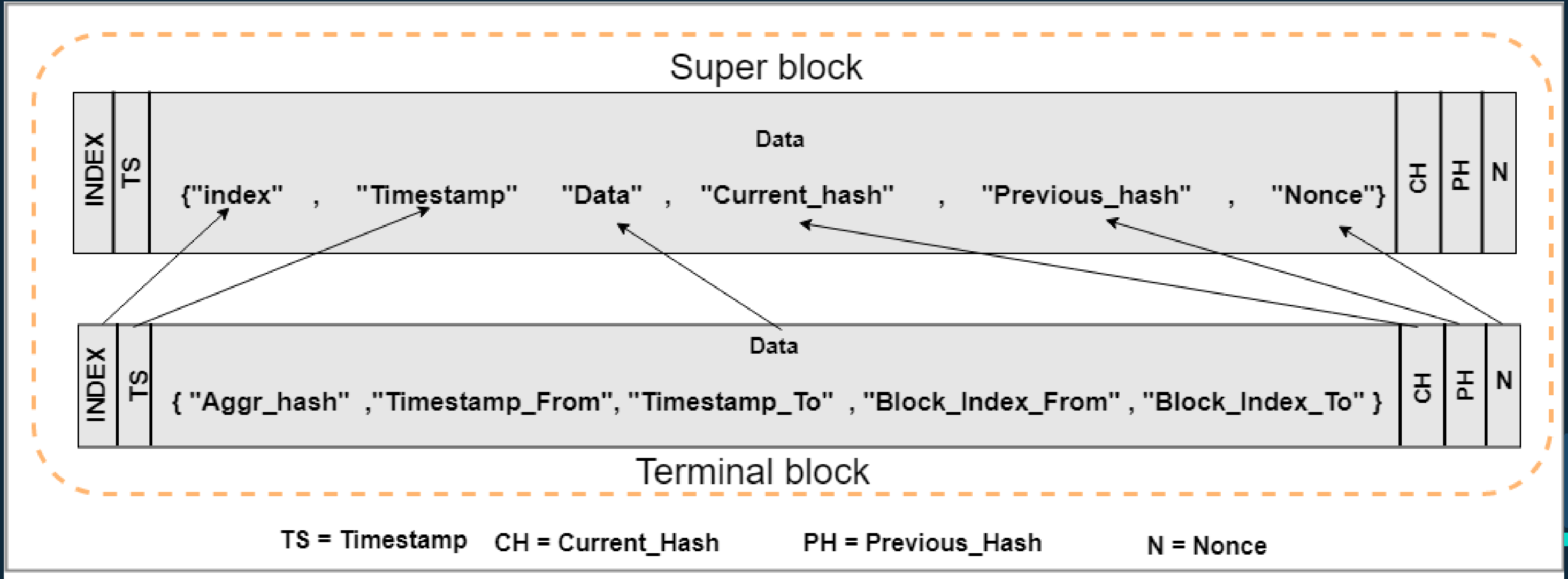
- Immutable Databases
- Log Unit
- Dynamic Configuration
- Reliability
- API
- Extension to a framework for secure log storage

Thank you
Q & A



Animation in this page: https://steemit.com/blockchain/@kid4life/can-a-blockchain-prevent-food-contamination-ideas2reality

Super Block and Terminal Block Relationship


Super block
INDEX
TS
Data
{"index" , "Timestamp" "Data" , "Current_hash" , "Previous_hash" , "Nonce"}
CH
PH
N
INDEX
TS
Data
{ "Aggr_hash" ,"Timestamp_From", "Timestamp_To" , "Block_Index_From" , "Block_Index_To" }
CH
PH
N
Terminal block

TS = Timestamp CH = Current_Hash PH = Previous_Hash N = Nonce